DL-ITQ-Practice

April 13, 2023

1 Classification

[1]:
```
import tensorflow as tf
print(tf.__version__)
```

```
2.12.0
```

[2]:
```
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
Num GPUs Available:  1
```

[ ]:
```
from tensorflow import keras as kr
print(kr.__version__)
```

```
2.12.0
```

[ ]:
```
fashion_mnist = kr.datasets.fashion_mnist
(x_train,y_train), (x_test, y_test) = fashion_mnist.load_data()
```

[ ]:
```
print(x_train.shape)
```

```
(60000, 28, 28)
```

[ ]:
```
x_valid, x_train = x_train[:5000]/255, x_train[5000:]/255
y_valid, y_train = y_train[:5000], y_train[5000:]
```

[ ]:
```
class_names = ["T-shirt/top",␣
↪"Trouser","pullover","Dress","Coat","Sendal","Shirt","Sneaker","Bag","Ankle␣
↪boot"]
```

[ ]:
```
class_names[y_train[6]]
```

[ ]:
```
'Coat'
```

[ ]:
```
model = kr.models.Sequential()
model.add(kr.layers.Flatten(input_shape =[28,28]))
model.add(kr.layers.Dense(300, name= "Hidden1", activation="relu"))
model.add(kr.layers.Dense(100, name= "Hidden2", activation="relu"))
model.add(kr.layers.Dense(10,name= "Output", activation="softmax"))
```

[ ]:
```
model.summary()
```

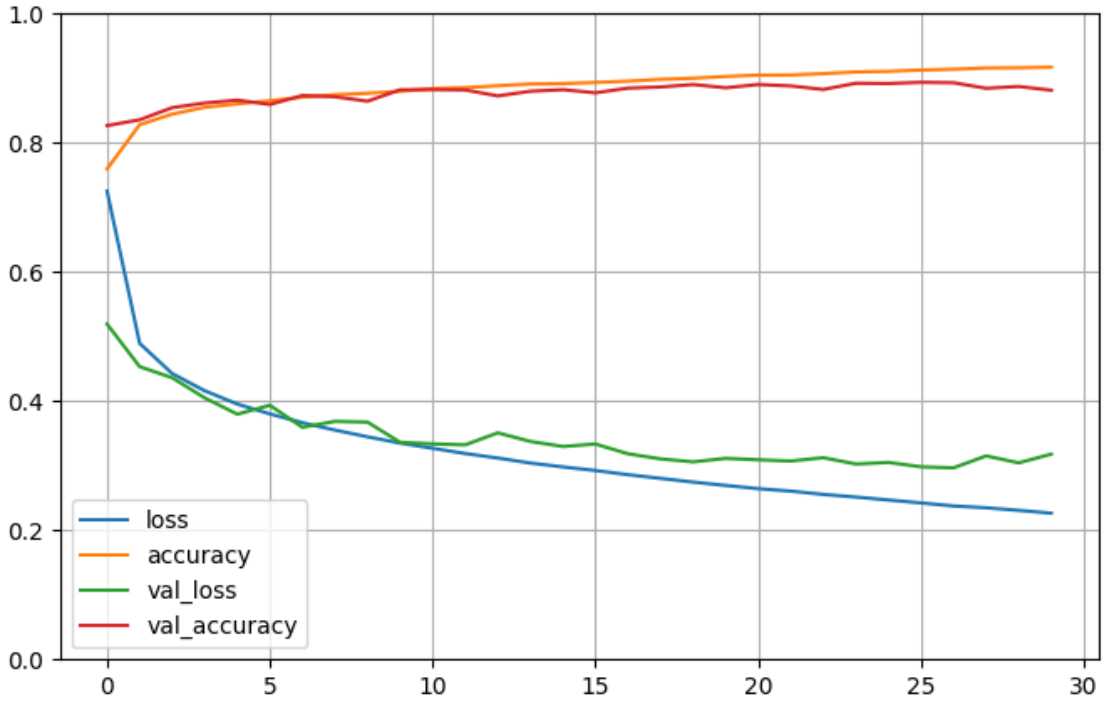
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 Hidden1 (Dense)             (None, 300)               235500

 Hidden2 (Dense)             (None, 100)               30100

 Output (Dense)              (None, 10)                1010

=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
model.compile(loss = "sparse_categorical_crossentropy", optimizer = "sgd",␣
 ↪metrics = ["accuracy"])
history = model.fit(x_train,y_train, epochs=30, validation_data =␣
 ↪(x_valid,y_valid))
```

```
Epoch 1/30
1719/1719 [==============================] - 10s 5ms/step - loss: 0.7231 -
accuracy: 0.7662 - val_loss: 0.5060 - val_accuracy: 0.8286
Epoch 2/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.4894 -
accuracy: 0.8291 - val_loss: 0.4678 - val_accuracy: 0.8352
Epoch 3/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.4454 -
accuracy: 0.8424 - val_loss: 0.4203 - val_accuracy: 0.8548
Epoch 4/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.4179 -
accuracy: 0.8533 - val_loss: 0.4018 - val_accuracy: 0.8602
Epoch 5/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3989 -
accuracy: 0.8592 - val_loss: 0.3725 - val_accuracy: 0.8754
Epoch 6/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3828 -
accuracy: 0.8652 - val_loss: 0.3768 - val_accuracy: 0.8688
Epoch 7/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3696 -
accuracy: 0.8695 - val_loss: 0.3671 - val_accuracy: 0.8718
Epoch 8/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3583 -
```

```
accuracy: 0.8732 - val_loss: 0.3509 - val_accuracy: 0.8774
Epoch 9/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3476 -
accuracy: 0.8767 - val_loss: 0.3615 - val_accuracy: 0.8764
Epoch 10/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3374 -
accuracy: 0.8802 - val_loss: 0.3473 - val_accuracy: 0.8768
Epoch 11/30
```

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.show()
```



[ ]:
```
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 58.6208 -
accuracy: 0.8525
```

[ ]:
```
[58.62080383300781, 0.8525000214576721]
```

[ ]:
```
print(model.predict(x_test[:3]))
print(y_test[3])
```

```
1/1 [==============================] - 0s 55ms/step
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
1
```

[ ]:
```
import numpy as np
y_pred = np.argmax(model.predict(x_test[:3]), axis=-1)
y_pred
```

```
1/1 [==============================] - 0s 12ms/step
```

[ ]:
```
array([9, 2, 1])
```

[ ]:
```
np.array(class_names)[y_pred]
```

[ ]:
```
array(['Ankle boot', 'pullover', 'Trouser'], dtype='<U11')
```

[ ]:
```
np.array(class_names)[y_test[:3]]
```

[ ]:
```
array(['Ankle boot', 'pullover', 'Trouser'], dtype='<U11')
```

2 Regresi

[ ]:
```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
housing =  fetch_california_housing()
housing
```

[ ]:
```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,
2.55555556,
           37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
           37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
           37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
           39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
           39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
           39.37      , -121.24      ]]),
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
```

```
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing
dataset\n--------------------------\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 20640\n\n    :Number of Attributes: 8 numeric, predictive
attributes and the target\n\n    :Attribute Information:\n        - MedInc
median income in block group\n        - HouseAge      median house age in block
group\n        - AveRooms      average number of rooms per household\n        -
AveBedrms     average number of bedrooms per household\n        - Population
block group population\n        - AveOccup      average number of household
members\n        - Latitude      block group latitude\n        - Longitude
block group longitude\n\n    :Missing Attribute Values: None\n\nThis dataset was
obtained from the StatLib
repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\nThe
target variable is the median house value for California districts,\nexpressed
in hundreds of thousands of dollars ($100,000).\n\nThis dataset was derived from
the 1990 U.S. census, using one row per census\nblock group. A block group is
the smallest geographical unit for which the U.S.\nCensus Bureau publishes
sample data (a block group typically has a population\nof 600 to 3,000
people).\n\nA household is a group of people residing within a home. Since the
average\nnumber of rooms and bedrooms in this dataset are provided per
household, these\ncolumns may take surprisingly large values for block groups
with few households\nand many empty houses, such as vacation resorts.\n\nIt can
be downloaded/loaded using
the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n.. topic::
References\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial
Autoregressions,\n      Statistics and Probability Letters, 33 (1997)
291-297\n'}
```

[ ]:
```
x_train_full, x_test, y_train_full, y_test = train_test_split(housing.data,␣
↪housing.target)
x_train, x_valid, y_train, y_valid = train_test_split(x_train_full, y_train_full)
```

[ ]:
```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_valid = scaler.transform(x_valid)
x_test = scaler.transform(x_test)
```

[ ]:
```
print(x_train.shape[1:])
```

```
(8,)
```

[ ]:
```
model_reg = kr.models.Sequential([
    kr.layers.Dense(30, activation = "relu", input_shape=x_train.shape[1:]),
    kr.layers.Dense(1)
])
```

[ ]:
```
opt = kr.optimizers.SGD(0.1, clipnorm=1.)
model_reg.compile(loss=["mse"], loss_weights=[0.9, 0.1], optimizer=opt)
history = model_reg.fit(x_train,y_train, epochs=20, validation_data =␣
 ↪(x_valid,y_valid))
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.SGD` runs
slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, located at
`tf.keras.optimizers.legacy.SGD`.
WARNING:absl:There is a known slowdown when using v2.11+ Keras optimizers on
M1/M2 Macs. Falling back to the legacy Keras optimizer, i.e.,
`tf.keras.optimizers.legacy.SGD`.

Epoch 1/20
363/363 [==============================] - 2s 5ms/step - loss: 0.4778 -
val_loss: 1.2084
Epoch 2/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3578 -
val_loss: 0.3476
Epoch 3/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3501 -
val_loss: 0.3867
Epoch 4/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3365 -
val_loss: 0.3255
Epoch 5/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3247 -
val_loss: 0.3042
Epoch 6/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3198 -
val_loss: 0.3543
Epoch 7/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3137 -
val_loss: 1.3766
Epoch 8/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3272 -
val_loss: 0.6769
Epoch 9/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3129 -
val_loss: 0.3528
Epoch 10/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3232 -
val_loss: 0.3652
Epoch 11/20
```

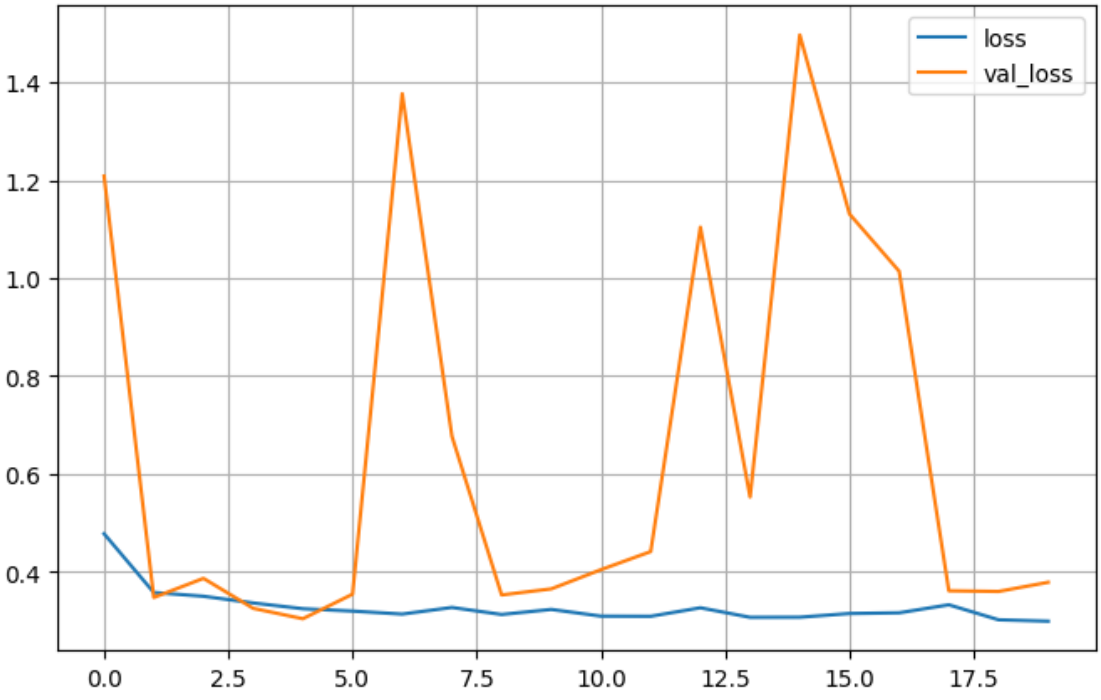
```
363/363 [==============================] - 2s 5ms/step - loss: 0.3095 -
val_loss: 0.4045
Epoch 12/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3092 -
val_loss: 0.4414
Epoch 13/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3266 -
val_loss: 1.1040
Epoch 14/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3071 -
val_loss: 0.5528
Epoch 15/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3074 -
val_loss: 1.4967
Epoch 16/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3149 -
val_loss: 1.1309
Epoch 17/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3163 -
val_loss: 1.0139
Epoch 18/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3327 -
val_loss: 0.3613
Epoch 19/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3021 -
val_loss: 0.3599
Epoch 20/20
363/363 [==============================] - 2s 5ms/step - loss: 0.2992 -
val_loss: 0.3787
```

[ ]:
```
model_reg.evaluate(x_test,y_test)
```

```
162/162 [==============================] - 0s 3ms/step - loss: 0.3178
```

[ ]:
```
0.317816823720932
```

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8,5))
plt.grid(True)

plt.show()
```


loss
val_loss
1.4
1.2
1.0
0.8
0.6
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5